AD 673442

OPERATIONS EVALUATION GROUP

COMPUTER CALCULATION OF DISCRETE FOURIER TRANSFORMS USING THE FAST FOURIER TRANSFORM

By J.C. Wilson

OEG Research Contribution No. 81

This Research Contribution does not necessarily represent the opinion of the Department of the Navy

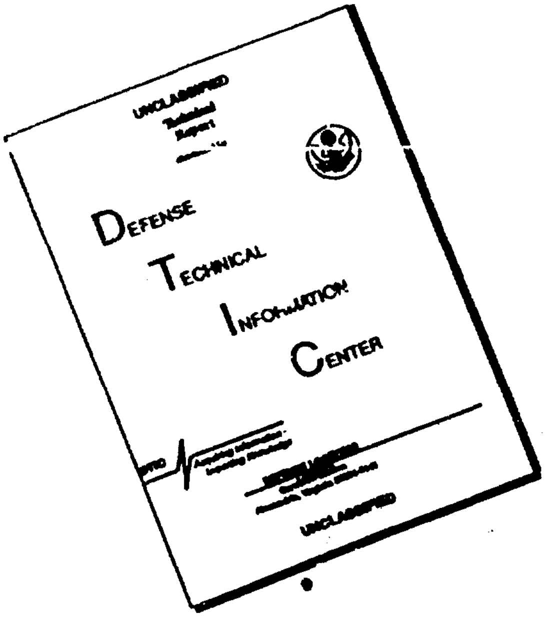
Distribution of this document is unlimited.

CNA
CENTER FOR NAVAL ANALYSES
OF THE UNIVERSITY OF ROCHESTER
1401 WILSON BOULEVARD ARLINGTON, VIRGINIA 22209

CONTRACT N00014-68-A-0091

20

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

CENTER FOR NAVAL ANALYSES

OF THE

UNIVERSITY OF ROCHESTER

1401 Wilson Boulevard
Arlington, Virginia 22209

Area code: 703
[illegible] 4-9400

(OEG)463-68
14 Aug 1968

From: Acting Director, Operations Evaluation Group
To: Distribution List

Subj: Operations Evaluation Group/CNA Research Contribution No. 81; forwarding of

Encl: (1) OEG RC #81, "Computer Calculations of Discrete Fourier Transforms Using the Fast Fourier Transform," Unclassified of 5 June 1968

1. Enclosure (1) is forwarded as a matter of possible interest.

2. This paper describes a computer program which determines the Discrete Fourier Transform of a set of data, using a recently developed technique known as the Fast Fourier Transform. The relation between Discrete Fourier Transform and Fourier Series when the data is periodic is also shown.

3. Research Contributions represent the opinion of the authors and are distributed for their possible value in further analysis and related problems. The enclosure has not been reviewed in detail and does not necessarily represent the opinion of the Center for Naval Analyses or the Department of the Navy.

4. The enclosure has been approved for public release.

5. Qualified users of the Defense Documentation Center may request additional copies from DDC.

ERVIN KAPOS
Acting Director
Operations Evaluation Group

DISTRIBUTION:
Attached List

OEG RESEARCH CONTRIBUTION NO. 81

Operations Evaluation Group

CENTER FOR NAVAL ANALYSES

COMPUTER CALCULATION OF DISCRETE FOURIER TRANSFORMS USING THE FAST FOURIER TRANSFORM

By J. C. Wilson

5 June 1968

Work conducted under contract N00014-68-A-0091

Enclosure (1) to
OEG ltr (OEG)463-68
Dated 14 August 1968

ABSTRACT

This research contribution describes a computer program (CNA Number 76-67) which determines the Discrete Fourier Transform of a set of data, using a recently developed technique known as the Fast Fourier Transform. The relation between Discrete Fourier Transforms and Fourier Series when the data is periodic is also shown.

(REVERSE BLANK)

COMPUTER CALCULATION OF DISCRETE FOURIER TRANSFORMS USING THE FAST FOURIER TRANSFORM

The Fast Fourier Transform (FFT) is a technique recently developed to compute the Discrete Fourier Transform (DFT) of a set of data. The Discrete Fourier Transform can be considered as a good approximation to the Fourier Series coefficients of a time series (for periodic phenomena) or to the Fourier Integral (for aperiodic phenomena) whenever the time series is known only at discrete points in time. In addition, the DFT is a useful transform in its own right and is discussed in some detail in the references. Interest in the Fast Fourier Transform was generated at CNA by some problems in message transmissions, which involved determining the delay between transmission of a message and any requests to service it, and also involved investigating possible periodicities in service requests.

Computing the DFT of a time series is very similar to computing Fourier Series coefficients and, until recently, has taken an amount of time approximately proportional to N^2, where N is the number of data points used. Computing the Fast Fourier Transform, however, uses time proportional to $N \log_2 N$, which can result in a substantial saving of computer time if the number of data points is large. An additional advantage is that the roundoff error is reduced because fewer calculations have to be performed.

Because of the speed of the FFT and the consequent greater utility of the DFT, a program (CNA 76-67) was written to compute the DFT of a set of data using the Fast Fourier Transform. This program can provide a method of detecting periodicities in data by using the DFT as an approximation to Fourier Series. In addition, the user who is more familiar with transform techniques can use it to determine relations between several sets of data (least squares fits, time lags, cross-correlation, etc.).

A complete description of the Fast Fourier Transform is too involved to be given here, but is presented quite clearly in references (a) through (e). Basically, the technique consists of successively dividing the data into groups of N/2, N/4, N/8, ..., N/N points, then recombining these groups with appropriate weighting factors.

Three major restrictions in the use of the program should be mentioned. First, the program works only for sets of $N=2^K$ data points, where K is an integer. In other words the user may have to sacrifice some of his data to bring the number of points down to the next lower power of 2, so that division by 2, 4, 8, ..., 2^K will be possible. Second, when the Discrete Fourier Transform is used as an approximation to the Fourier series of a set of data, only the first N/2 DFT coefficients, instead of all N, should be considered. The reason for this is discussed in appendix A. And finally, because of a limitation of the CDC 3400 FORTRAN system, arrays of data points and coefficients are from 1 to N instead of from 0 to N-1 (as used in the equations

in the derivations), which will usually entail some extra input and output operations to keep the format in line with the format in the derivations.

Appendix A contains the definition of the Discrete Fourier Transform and its relation to the Fourier Series. Appendix B is a description of the computer program, which uses the FFT technique, along with its limitations and some possible uses. Also included is a listing of the program itself, which is written in FORTRAN II for the CDC 3400 computer. Appendix C gives an example of how the program can be used and some numerical results.

APPENDIX A

APPENDIX A

DEFINITIONS

Given a time series $x(k\Delta t)$, $k=0, \ldots, N-1$, the Discrete Fourier Transform (DFT) and its inverse are defined as follows:

$$B(n)=\frac{1}{N}\sum_{k=0}^{N-1} x(k)\exp\left(\frac{-2\pi ikn}{N}\right), \qquad n=0, 1, \ldots, N-1 \tag{1a}$$

and

$$x(k)=\sum_{n=0}^{N-1} B(n)\exp\left(\frac{+2\pi ink}{N}\right), \qquad k=0, 1, \ldots, N-1 \tag{1b}$$

where $i = \sqrt{-1}$

$B(n)$ will in general be complex when x is real. The similarity between the DFT and the Fourier Series is evident when we consider the exponential form of the Fourier Series:

$$C(n)=\frac{1}{T}\int_0^T x(t)\exp\left(\frac{-2\pi int}{T}\right)dt, \qquad \text{all integer } n, \tag{2a}$$

and

$$x(t)=\sum_{n=-\infty}^{n=\infty} C(n)\exp\left(\frac{+2\pi int}{T}\right), \qquad \text{where } x(t) \text{ is piecewise continuous and periodic with period } T. \tag{2b}$$

Now if the above integral is approximated by its Riemann sum, $C(n)$ becomes approximately

$$C(n)\doteq\frac{1}{N}\sum_{k=0}^{N-1} x(k\Delta t)\exp\left(\frac{-2\pi ink}{N}\right),$$ where $T=N\Delta t$. In other words we use N samples in the semi-open interval $[0, T)$.

But the sum on the right is just $B(n)$, so we have

$$C(n)\doteq B(n) \tag{3}$$

This then is the relation between Fourier Series and Discrete Fourier Transforms when $x(t)$ is periodic. The $C(n)$ tell us how much of $x(t)$ can be attributed to sinusoids of "frequency"

$$f=\frac{n}{T}=\frac{n}{N}\left(\frac{1}{\Delta t}\right),$$

and can be approximated by the DFT coefficients $B(n)$. This becomes clearer when we realize that in fact (reference (b))

$$B(n)=\sum_{j=-\infty}^{\infty} C(n+jN), \qquad n=0, 1, \ldots, N-1. \tag{4}$$

That is, $B(n)$ is the sum of overlapped segments of $C(n)$. Figure A-1 shows this relationship between $B(n)$ and $C(n)$. In order to make $B(n)$ a better approximation to $C(n)$ we must increase the number of samples in the period T.

Now if $x(k)$ is real, $C(n)$ will be an even function of n; that is, $C(n)=C(-n)$. Equation (4) then gives us some further information about $B(n)$:

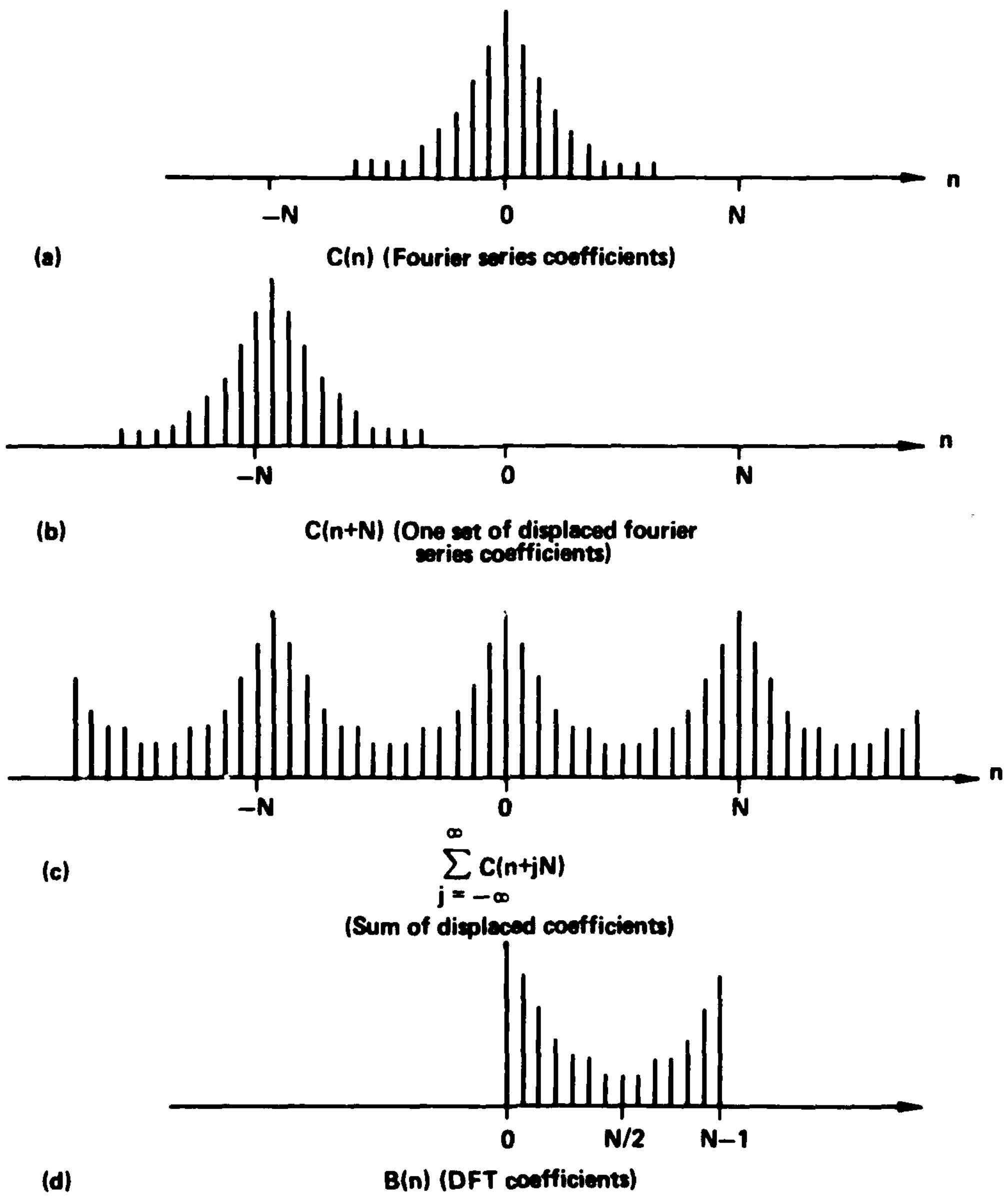


FIG. A–1: SHOWING HOW THE DFT COEFFICIENTS ARE RELATED TO THE FOURIER SERIES COEFFICIENTS OF A TIME SERIES

$$B(N-n)=\sum_{j=-\infty}^{\infty} C(-n+jN+N) = \sum_{j=-\infty}^{\infty} C(-n+(j+1)N)$$

$$= \sum_{m=-\infty}^{\infty} C(-n+mN) \quad \text{with } m=j+1$$

$$= \sum_{m=-\infty}^{\infty} C(n-mN) \quad \text{since C is even}$$

$$= \sum_{k=\infty}^{-\infty} C(n+kN) \quad \text{with } k=-m$$

$$= \sum_{k=-\infty}^{\infty} C(n+kN) \quad \text{since the order of summation is immaterial}$$

$$B(N-n)=B(n) \qquad n=0, 1, \dots, N-1 \tag{5}$$

Thus B(n) is symmetrical about n=N/2 (see figure A-1). Care must be taken, therefore, not to use B(n) as an approximation to C(n) when $n \geq N/2$.

APPENDIX B

APPENDIX B

DESCRIPTION OF THE PROGRAM

The core of the program to compute the DFT of a time series is really quite compact, and can be expressed as a set of 4 nested do loops:

```
      DO 100 j=1, s
      DO 100 k=0, 2^(s-j)-1
      DO 100 n=0, 2^(j-1)-1
      DO 100 m=0, 1
```

$$100 \qquad B(n+k\cdot 2^{j}+m\cdot 2^{j-1})=B(n+k\cdot 2^{j})+(-1)^{m}B(n+k\cdot 2^{j}+2^{j-1})\exp\left[\frac{-2\pi i}{2^{s}}\,n(s+1-j)\right]$$

where $N=2^{s}$ data points are used.*

An advantage of the program not mentioned in the body of the text is that the coefficients can use the same storage as the original data, since the program exchanges pairs of values (m=0, 1 in the above formulation) after appropriately weighting them. An important programming consideration, however, is that all arrays must be in complex notations to effect this space-saving, so real time series must be converted to a complex format before being used.

The program allows up to $1024=2^{10}$ data points. To store up to 2^{15} points, the user need only redimension the arrays to set aside that much storage. If still more data is used, then complicated changes must be made to the subroutine written in COMPASS.

The user may also call for the inverse transform to get back a time series from a set of coefficients by calling the subroutine INVERSE. At any time after calling the subroutines for the transform or inverse transform, the results are stored (in complex form) in the original data locations, available to the user for printout or manipulation.

As a final feature, the user may call subroutines to smooth the coefficients. The reason for wanting to smooth the DFT coefficients is that our data extends only over a finite time interval; this, however, usually only represents the portion of the process we have chosen to record, and in fact, the process will often be infinite in duration. Using only that data we have recorded is equivalent to clipping the actual process at arbitrary end points in time. In the frequency domain this distorts the frequency components (DFT coefficients) from what they would be if we were to consider the process as having infinite duration in time. To reduce this distortion, the data points can be smoothed so that the clipping is not so pronounced. The drawback, however, is that some of the statistical value of the data is lost, since smoothing distorts the data in the time domain. Thus, it is wise to look at both the unsmoothed coefficients as well as the smoothed coefficients in any practical problem. In the program described in this paper, the smoothed results may either be simply printed out (CALL SMOOTH1) or placed in the data cells (CALL SMOOTH).

The program is available in the form of subroutines assembled in a binary deck. It is the option of the user to plot the data and to make his own printouts.

---

*A limitation of the CDC 3400 computer is that the index on the array B must run from 1 to N instead of from 0 to N-1. Therefore, in the actual program, B(j) is the DFT coefficient of frequency $(j-1)/N\Delta t$ instead of $j/N\Delta t$.

SUMMARY OF AVAILABLE SUBROUTINES

B(n), n=1, ..., N is assumed to be a complex array with $N=2^M$ elements.

XFORM(M, B) computes the 2^M DFT coefficients of the series B(1), B(2), ..., B(N) and stores these coefficients in the array B, replacing the original series.

INVERSE(M, B) computes the 2^M inverse DFT coefficients of the series B(1), B(2), ..., B(N) and stores these coefficients in the array B, replacing the original series.

SMOOTH(M, B) smooths the 2^M DFT coefficients located in the array B, replacing the elements of B with these smoothed coefficients.

SMOOTH1(M, B) smooths the 2^M DFT coefficients located in the array B and prints the smoothed coefficients. The original coefficients are left undisturbed.

```
      S[illegible] XFORM(M,H)
      T[illegible]X H, W, HONE, HTWO, CM[illegible] [illegible]EXP
      DIMEN[illegible]1024), W(512)
      COMMON W
       MPOW=2**M
        [illegible]1, MPOW
        [illegible]
      CA[illegible]SE(JR,M)
      IF (JR[illegible]) 5, 25, 25
  5   [illegible]
       [illegible]
      H(JR+1)=HONE
  [illegible] CON[illegible]
      MPOW=MPOW/2
   1[illegible] DO 35 [illegible], MPOW
       HONE=CMPLX(0.,-3.14159265360*(J-[illegible])/MPOW)
      W([illegible]EXP(HONE)
  3[illegible]
C     COMPUTE TRANSFORM
      MZE[illegible]
        [illegible]100 J=1,M
          JA=2**(J-1)
          JA1=JA-1
          JY=2*JA
          JZ=2**(M-J)
          JZ1=JZ-1
      DO 80 K=MZERO,JZ1
      DO 80 L=MZERO,JA1
          LA=L*(M+1-J)+1
      HONE=H(L+1+K*JY)+H(L+1+K*JY+JA)*W(LA)
      HTWO=H(L+1+K*JY)-H(L+1+K*JY+JA)*W(LA)
      H(L+1+K*JY    )=HONE
      H(L+1+K*JY+JA)=HTWO
   80 CONTINUE
  100 CONTINUE
      MPOW=2**MPOW
      DO 110 J=1,MPOW
  11[illegible] H(J)=H(J)/MPOW
      RETURN
      END

      SUBROUTINE INVERSE(M,H)
      TYPE COMPLEX H, W, HONE, HTWO, CMPLX, CEXP
      DIMENSION H(1024), W(512)
      COMMON W
       MPOW=2**M
       DO 30 J=1, MPOW
       JR=J-1
      CALL REVERSE(JR,M)
      IF (JR+1-J) 30, 25, 25
   25 HONE=H(J)
       H(J)=H(JR+1)
      H(JR+1)=HONE
   30 CONTINUE
      MPOW=MPOW/2
   1[illegible] DO 35 J=1,MPOW
       HONE=CMPLX(0.,-3.14159265360*(J-1.)/MPOW)
      W(J)=CEXP(HONE)
   35 CONTINUE
C     COMPUTE TRANSFORM
      MZERO=0
      DO 100 J=1,M
          JA=2**(J-1)
          JA1=JA-1
          JY=2*JA
          JZ=2**(M-J)
          JZ1=JZ-1
      DO 80 K=MZERO,JZ1
      DO 80 L=MZERO,JA1
          LA=L*(M+1-J)+1
      HONE=H(L+1+K*JY)+H(L+1+K*JY+JA)*W(LA)
      HTWO=H(L+1+K*JY)-H(L+1+K*JY+JA)*W(LA)
      H(L+1+K*JY    )=HONE
      H(L+1+K*JY+JA)=HTWO
   80 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SMOOTH(K,B)
      TYPE COMPLEX B, BEG, XONE, XTWO
      DIMENSION B(1024)
       KPOW=2**K
      BEG=B(1)
      XTWO=.25*(-B(2)+2.*B(1)-B(KPOW))
      KPOWM=KPOW-2
      DO 5 J=1,KPOWM
      XONE=XTWO
      XTWO=.25*(-B(J)+2.*B(J+1)-B(J+2))
      B(J)=XONE
    5 CONTINUE
      XONE=XTWO
      XTWO=.25*(-B(KPOW-1)+2.*B(KPOW)-BEG)
      B(KPOW-1)=XONE
      B(KPOW)=XTWO
      RETURN
      END
```

```
      SUBROUTINE SMOOTH1(K,B)
      TYPE COMPLEX B, BS
      DIMENSION B(1024)
      PRINT 199
      PRINT 200
      KPOW=2**K
      BS=.25*(-B(2)+2.*B(1)-B(KPOW))
      ABL=CABS(BS)
      JL=0
      AFL=1.
      PRINT 201, JL, AFL, ABL, BS
      KPOWM=KPOW-1
      DO 30 J=2, KPOWM
      JL=J-1
      BS=.25*(-B(JL)+2.*B(J)-B(J+1))
      AEL=CABS(BS)
      AFL=AEL/ABL
      PRINT 201, JL, AFL, AEL, BS
   30 CONTINUE
      BS= .25*(-B(KPOWM)+2.*B(KPOW)-B(1))
      AEL=CABS(BS)
      AFL=AEL/ABL
      PRINT 201, KPOWM, AFL, AEL, BS
  199 FORMAT (1H1,* SMOOTHED FOURIER COEFFICIENTS*,////,* REAL COEF = *
     1*REAL PART OF COEF*,/,* IMAG COEF = IMAG PART OF COEF*,/,
     2* ABS VALUE = ABS VALUE OF COEF*,/,* ADJ COEF =  ABS VALUE *
     3*DIVIDED BY  ABS VALUE OF COEF AT ZERO*,//)
  200 FORMAT (1X,*FREQ*,3X,*ADJ COEF*,3X,*ABS VALUE*,3X,
     1    *REAL COEF*,3X,*IMAG COEF*,/)
  201 FORMAT (1X,I4,3X,F8.5,3X,E9.3,3X,C(E9.2,E12.2))
      RETURN
      END
```

```
                                      IDENT   REVERSE     ROUTINE TO REVERSE BITS
00000       00017           SAVE      BSS     15          1 THRU N OF WORD J.
00017 0000000000000000      ZERO      OCT     0           ENTER WITH ARGS (J,N)
00020 0000000000000001      MASK      OCT     1
                                      ENTRY   REVERSE
00021 75 0 77777            REVERSE   SLJ     **
      56 1 00042  P                   SIU     2,1         SAVE IRS
00022 57 2 00042  P                   SIL     2,2         1 AND 2
      52 1 00021  P                   LIU     REVERSE,1   LOAD ADDRESS OF ARGS
00023 12 1 00000                      LDA     0,1         PUT IN A
      60 0 00027  P                   SAU     LOAD        STORE ADD. OF N
00024 01 0 00030                      ARS     24
      60 0 00031  P                   SAU     J           STORE ADD. OF J
00025 61 0 00041  P                   SAL     END
      51 1 00001                      INI     1,1
00026 56 1 00021  P                   SIU     REVERSE,1   PUT IN PROPER RETURN ADD.
      50 0 00000
00027 12 0 77777            LOAD      LDA     **          BRING IN N
      11 0 77776                      INA     -1          SUB 1
00030 80 7 40542                      ATI     2           IN IR 2
      50 0 00000
00031 12 0 77777            J         LDA     **          BRING IN J
      16 0 00020  P                   LDQ     MASK        LOAD MASK
00032 50 1 00000                      ENI     0,1         PUT ZERO IN IR1
      50 0 00000
00033 47 1 00000  P         LOOP      STL     SAVE,1      PUT 0 OR 1 IN VALUES
      01 0 00001                      ARS     1           OF SAVE, CORRESPONDING
00034 51 1 00001                      INI     1,1         TO RIGHTMOST BIT
      55 2 00033  P                   IJP     LOOP,2      FINISHED
00035 10 0 00000                      ENA     0
      50 2 00000                      ENI     0,2
00036 51 1 77776                      INI     -1,1
      50 0 00000
00037 05 0 00001            AD        ALS     1
      14 2 00000  P                   ADD     SAVE,2
00040 51 2 00001                      INI     1,2
      50 0 00000
00041 55 1 00037  P         END       IJP     AD,1
      20 0 77777                      STA     **
00042 50 1 77777            Z         ENI     **,1
      50 2 77777                      ENI     **,2
00043 75 0 00021  P                   SLJ     REVERSE
      50 0 00000
                                      END
```

APPENDIX C

APPENDIX C

AN EXAMPLE

In order to show how the program works, we shall compute the Discrete Fourier Transform of $x(t)=\sin(2\pi ft)=\sin(2\pi t/4)$.

Let

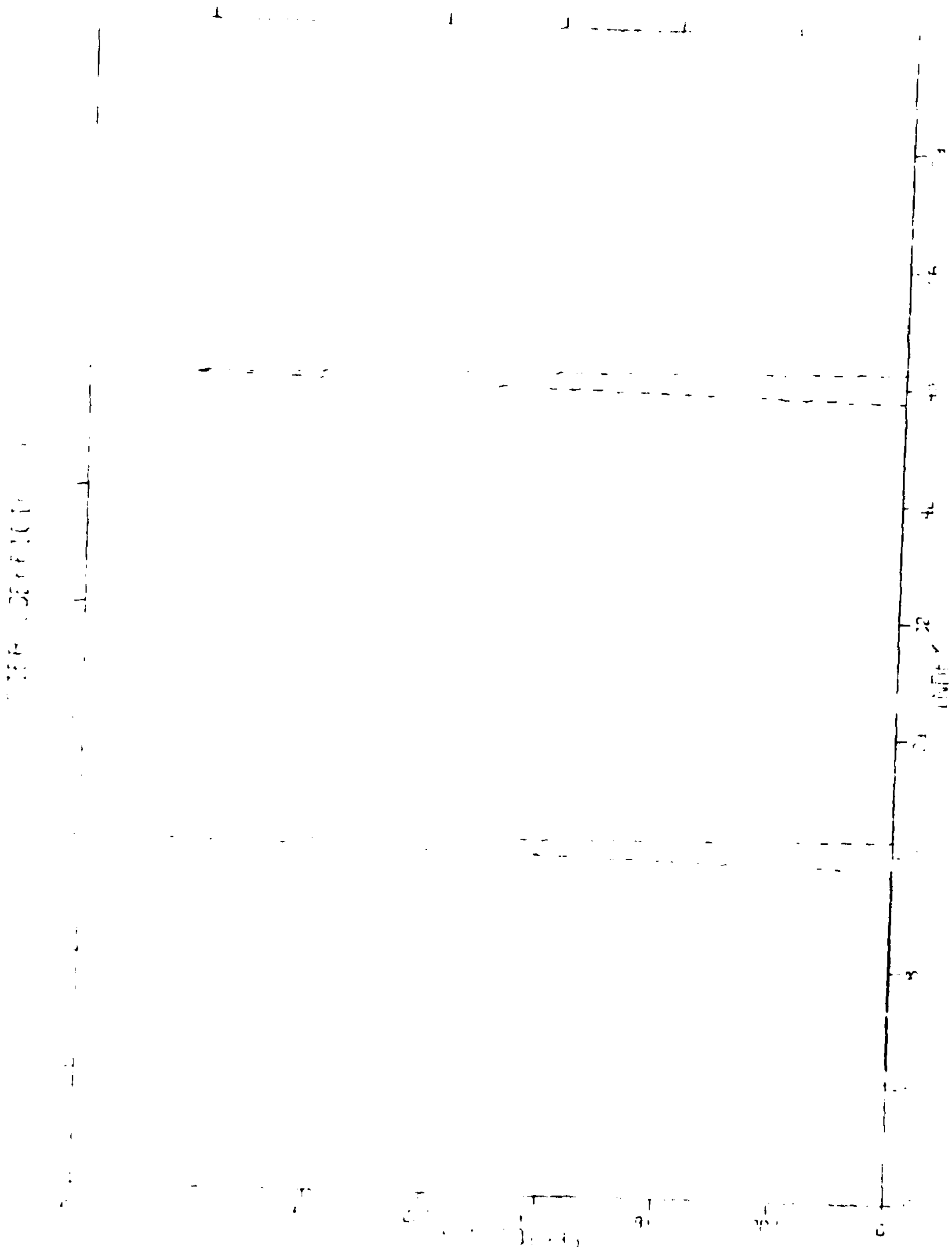
$N=64$ data points
$T=64$
so $\Delta t=1$

The Fourier <u>Series</u> representation of this function is a pair of spikes at $n=+16$ and $n=-16$, corresponding to a sine wave of frequency $f=\frac{n}{T}=\frac{16}{64}=\frac{1}{4}$. Sampling x every $\Delta t=1$ seconds gives

$$x(n)=\sin(2\pi n/4).$$

The modulus of each of the DFT coefficients of $x(n)$ is plotted in the accompanying graph. Note that there is indeed a peak at $n=16$, but as warned in appendix A, there is also a peak at $n=64-16=48$. This shows that we can only use the first $N/2$ coefficients when trying to detect periodicities. When using the DFT as a transform in its own right, this restriction does not necessarily hold.

```
      PROGRAM TEST
      TYPE COMPLEX A, CMPLX
      DIMENSION A(100), B(100), X(100)
C
C     ARRAY A MUST BE IN COMPLEX FORM FOR USE IN SUBROUTINE XFORM
      DO 50 J=1, 64
      X(J)=(J-1)*1.
   50 A(J)=CMPLX(SINF(2.*3.14159265*(J-1)/4.),0.)
C
C     N=6, SINCE 2**6=64
      CALL XFORM(6,A)
C     ARRAY A NOW CONTAINS THE DFT COEFFICIENTS
C
C     WE COMPUTE AND PLOT THE ABSOLUTE VALUE OF THE DFT COEFFICIENTS
      DO 60 J=1,64
   60 B(J)=CABS(A(J))
      CALL PLOTTER(X,B,64,-16,5HINDEX,5,12HCOEFFICIENTS,12,
     1 20HFOURIER COEFFICIENTS,20,0)
      END
```



References: (a) Cochran et al., "What is the Fast Fourier Transform?" IEEE Proceedings, pp. 1664-1674, Oct 1967

(b) Cooley, Lewis, and Welch, "Application of the Fast Fourier Transform to Computation of Fourier Integrals, Fourier Series, and Convolution Integrals," IEEE Transactions on Audio and Electroacoustics, pp. 79-84, Jun 1967

(c) Brigham and Morrow, "The Fast Fourier Transform," IEEE Spectrum, pp. 63-70, Dec 196[illegible]

Note: The June 1967 issue of the IEEE Transaction on Audio and Electroacoustics is devoted to applications of the Fast Fourier Transform and Discrete Fourier Transform.

None

Security Classification

DOCUMENT CONTROL DATA - R & D

Operations Evaluation Group, Center for Naval Analyses of the University of Rochester

2a. REPORT SECURITY CLASSIFICATION: None

2b. GROUP: None

Computer Calculation of Discrete Fourier Transforms using the Fast Fourier Transform

DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Contribution - 5 June 1968

AUTHOR(S) (First name, middle initial, last name)

Wilson, J.C.

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
5 June 1968	14	3

CONTRACT OR GRANT NO.: N00014-68-A-0091

PROJECT NO.: -----

9a. ORIGINATOR'S REPORT NUMBER(S): Operations Evaluation Group Research Contribution No. 81

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report): None

DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

SUPPLEMENTARY NOTES: None

12. SPONSORING MILITARY ACTIVITY: Office of Naval Research, Department of the Navy, Washington, D.C. 20350

ABSTRACT

This research contribution describes a computer program (CNA Number 76-67) which determines the Discrete Fourier Transform of a set of data, using a recently developed technique known as the Fast Fourier Transform. The relation between Discrete Fourier Transforms and Fourier Series when the data is periodic is also shown.

None
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Fast Fourier Transform						
Discrete Fourier Transform						
Fourier Series						
Fourier Transform						
Periodicities						

DD FORM 1473 (BACK)

None
Security Classification